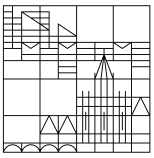
Task Sheet 8

Universität Konstanz



Solving the exploration task with an ANN

Deadline 01:00pm June 26th, 2024

Review June 26th, 2024

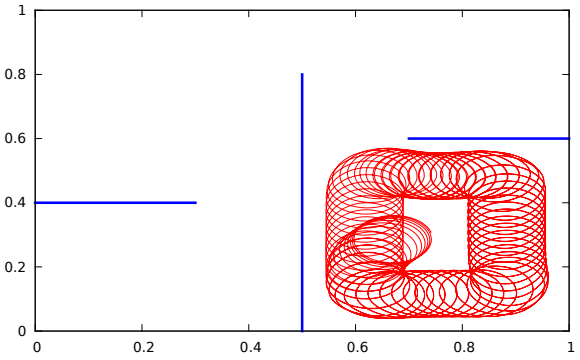
Lecture: *Evolutionary Robotics*, Summer Term 2024

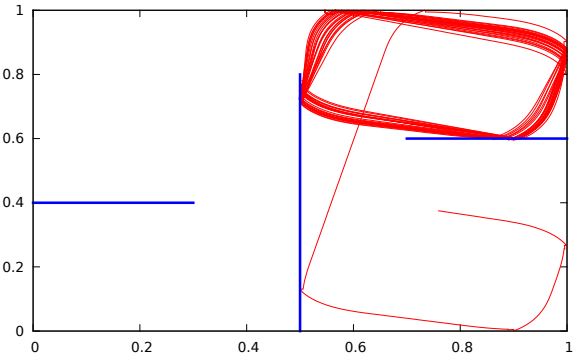
Lecturer: Prof. Dr.-Ing. Heiko Hamann

Tutor: Eduard Buss

Objectives:

- influence of bias neurons and other parameters
- avoid unwanted behaviors and push towards desired behaviors
- influence of initial positioning of the robot





Finally we get back to our robot simulator. The task is the same as in tutorial 2, task 2: exploration. For now place the robot at a fixed position with fixed heading initially in each evaluation as before. This time the robot controller is implemented as an ANN with hidden layer. Implement an ANN with 3 input neurons, 2 neurons in the hidden layer, and 2 output neurons (one for each wheel). In addition, implement an evolutionary algorithm of your choice or reuse your code. Try to make good decisions on the length of evaluations, the population size, etc. The fitness is again based on visited grid cells. This time, use a grid with high resolution of at least 100×100.

1. Do a number of independent evolutionary runs, log best and average fitness, and plot them.
2. Experiment a little with adding/deleting bias neurons that allow to shift the input from the sensors. Does that change the resulting behaviors qualitatively? What about the initialization of weights or the maximal angular velocity of the robot?
3. Maybe most of your evolved robots go in spirals/circles. Try to add a component to your fitness function that punishes too intensive turning behavior. In turn, if your robots tend to go in straight lines, try to make them go in spirals. If your robots tend to move close to walls, try to make them keeping a bigger distance. Try to tweak your fitness function (you can also experiment with different weighting of the components in the fitness function) until you see the desired change in the robots' behaviors.

4. Finally, place the robot at a random position and with random heading for each evaluation. This changes the whole approach qualitatively as we now have a non-deterministic fitness function (i.e., evaluating the very same ANN twice might give two different fitness values). Do a number of independent evolutionary runs, log best and average fitness, and plot them. What has changed in comparison to the deterministic approach? You can also test your best evolved controller started from different positions after evolution has been done.

Your submission:

- please zip your submission in a single file named: 'evoRobo_sheet1_YOURLASTNAME1_YOURLASTNAME2.zip'
- a readme file with the full names of all group members, a list of the tasks and subtasks you have completed and/or a list of tasks/subtasks you have not completed
- accepted file formats for plots: png and jpg
- all your code
- a plot of a typical robot trajectory of a robot controller evolved in a successful run for the deterministic approach (robot starts in exact same pose all the time)
- a plot of best fitness and population-average fitness over generations for a successful evolutionary run for the deterministic approach
- plot robot trajectories that indicate a qualitative change due to introducing bias neurons
- describe changes youve made to the fitness function and descrive/plot the observed/resulting robot trajectories
- a plot of best fitness and population-average fitness over generations for a successful evolutionary run for the non-deterministic approach (robot starts in random pose)
- a description of what you observed once the robot was initialized randomly at the beginning of each evaluation
- a plot of a robot trajectory for a robot placed in a completely different part of the arena that was evolved with random initilizations